SCHEDAE INFORMATICAE
VOLUME 20 (2011) DOI 10.4467/20838476SI.11.005.0291

A Genetic Algorithm for Clustering, Finding the Number of Clusters

Piotr Śmigielski
Institute of Computer Science, Jagiellonian University,
Łojasiewicza 6, 30-348 Kraków, Poland
e-mail: *smigielski.piotr@gmail.com*

Abstract. In this paper a genetic algorithm for clustering is proposed. The algorithm is based on the variable length chromosomes and the notion of local points density in the clustered set. Its role is to identify the number of clusters in the clustered set and to partition this set into particular clusters. The tests were conducted for two different sets of two dimensional data. The algorithm performed well in both cases. The tests presented the ability of the algorithm to partition the subsets combined with the thin dense area into separate clusters.

Keywords: genetic algorithm, clustering, variable length chromosome, cube.

1. Introduction

Soft computing methods are widely used for solving a problem of clustering [3, 6, 7, 8]. There is a couple of works presenting the utilization of genetic algorithms for clustering problems. An interesting approach was presented in [8]. That method operates directly on a schemata to represent those elements of the clustered set that are not assigned to a specific cluster in the particular iteration. The other methods such as in [7] operate on fully specified chromosomes which define membership of each element of the data set to the particular cluster. That method utilizes a straightforward crossing-over of the chromosomes of similar length. In the following paper a different approach is proposed. The notion of a size and placement of a cluster in the space is used. The chromosomes define boundaries of the cluster without directly specifying which cluster each element of the clustered set belongs to. Different numbers of clusters encoded by the chromosome and, in the same time, their different lengths involve specific methods of mutation and crossing-over.

2. The algorithm description

The aim of the algorithm is to localize dense areas (identified as the separate clusters) separated by the areas of significantly lower density of the points in the $\mathbb{R}^d$ space. The algorithm leads to the placement of the cubes in $\mathbb{R}^d$. Thus, they cover the points of the dense regions of the clustered set. We say that the cube is well placed when it covers as many points as possible while having the comparably small volume (see Sect. 2.4. The adaptation function).

The notion of the cube defined in that way allows to treat its center as the center of the cluster. At the same time, the number of cubes that were found defines the number of the clusters in the data set. The cubes can also be treated as defining particular clusters with their boundaries. In that case all of the points located inside the cube constitute one cluster, while points outside the cube are part of the other cluster or the noise.

2.1. The chromosome structure

The chromosome defines one, potential solution for the clustering problem. It specifies the location and size (volume) of the cubes that define clusters. It is logically divided into separate parts (sections), of which each one defines a single cube (cluster). The order of the sections is arbitrary, which makes the encoding redundant (each permutation of the sections in a chromosome defines the same clustering). The single cube is defined by its diagonal, which is expressed by the coordinates of its edges (two opposite apices of a cube). In $\mathbb{R}^2$ it will be a diagonal of a rectangle, while in $\mathbb{R}^3$ a diagonal of a rectangular prism. Such encoding is very memory efficient. The space complexity for one chromosome is $O(dk)$, where k is the number of sections (the number of the clusters found) and d is the dimension of the domain of a clustered set. The length of the chromosome is given by the formula $l = 2dk$. Formally, a chromosome can be interpreted as a vector in $\mathbb{R}^l$: $\left(x^1_{1_1}, ..., x^1_{1_d}, x^1_{2_1}, ..., x^1_{2_d}, x^2_{1_1}, ..., x^2_{1_d}, x^2_{2_1}, ..., x^2_{2_d}, ..., x^k_{1_1}, ..., x^k_{1_d}, x^k_{2_1}, ..., x^k_{2_d}\right)$.

This encoding does not require to store information about adherence of the points of the clustered set to a particular cluster. Instead, it defines the frames, adjusted during algorithm execution, which divide the data set.

As a solution for a clustering problem defining one cluster is trivial and not satisfying, it is assumed that a chromosome defines at least two sections – see Fig. 1. The algorithm operates on chromosomes of variable lengths and each operation supports that assumption. That operations are variable length crossing-over and mutation that also can change the chromosome length.

Each gene defines a single coordinate in $\mathbb{R}^d$ space. The genes are encoded with real numbers, which leads to limitation of computational overhead for decoding and encoding to the binary form.

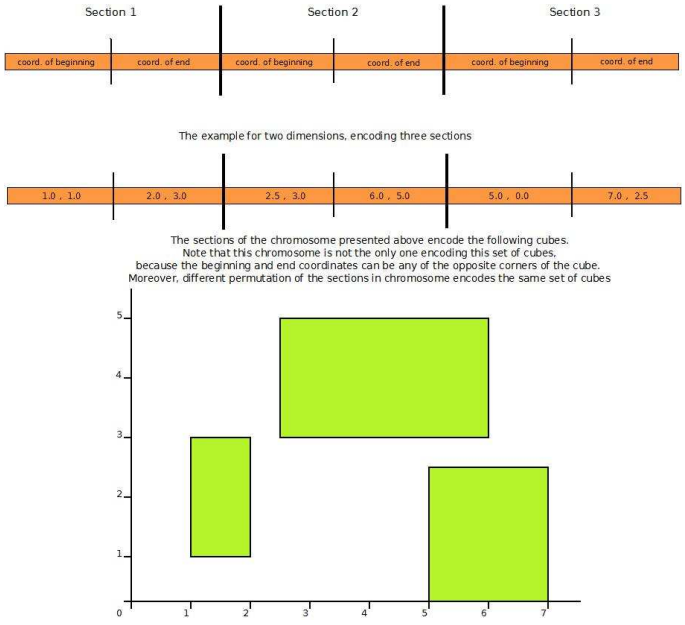

Fig. 1. The example of the chromosome divided into sections

2.2. Selection and crossing-over

The crossing-over is combined with the process of selecting the members of the population. The first step involves calculation of a value of the adaptation for each member of the population. For single crossing-over operation two chromosomes are randomly picked (without returning) using the roulette method. In this algorithm the value of the adaptation of one chromosome divided by the adaptation of the whole population (sum of adaptations of all members) is taken into account. Such a normalized value expresses the part of the roulette attributed to the chromosome. The bigger the part the greater is the chance for the chromosome to be chosen to take part in crossing-over. Here is the function for calculating the normalized value of the chromosome adaptation used in the roulette method $R(c) = \frac{A(c)}{\sum_{i=1}^{m} A(c_i)}$, where $A(c)$ is the adaptation of the chromosome c whereas m is the number of elements in the population.

The idea of crossing-over in this method is that a couple of the picked chromosomes produces two new members, of which a better one is chosen for the descendant population. With that chromosome, the better adapted parent also goes to a new population. That operation takes place $\frac{m}{2}$ times. Hence the requirement for the cardinality of the population is to be an even number. This method is patterned on a one of the first niching-like methods, proposed in 1970 by D.J. Cavicchio [4]. He introduced the method called pre-selection in which worse of the parents is substituted by the better of the children if its adaptation is higher than that of the parent.

With that method even small and medium- sized populations (even having around 20 members) reveal better ability to retain diversity with successive generations. It is because the sequence of genes, in a new population, is substituted by the one that is similar (a descendant replaces his parent) [5].

The crossing-over in the described algorithm can operate on a pair of chromosomes (C_1, C_2) of different lengths and produces one individual with the number of sections within the range $[2, L(C_1) + L(C_2) - 1]$, where L is the function returning the number of sections in a chromosome. The lower boundary comes from the assumption that the chromosome should encode at least 2 sections (and define at the same time at least two clusters).

The following list presents the steps of the crossing-over method in the described algorithm. It is assumed that there are two parent chromosomes already selected from the population. First of them is called a leader and the second one, a slave.

1. Randomly choosing the value g with the standarized normal distribution (0,1) within the range $(-L(C_{sl}) + 1, L(C_{sl}) - 1)$. Value s is calculated as g transformed with the operator $f(x) = \lfloor |x| \rfloor$. This value defines the shift between two selected chromosomes counted in full sections. The shift calculated in genes is equal to $2sd$. The range $[2, L(C_1) + L(C_2) - 1]$ results in the upper bound for s value as to be $L(C_{sl}) - 2$ which means that overlapping the part of chromosomes is at least two sections. The final state after shifting the chromosomes is presented in Fig. 2. It is worth mentioning that the prefix always belongs to the slave chromosome (it exists if $s \neq 0$), whereas the suffix can belong to either the slave or the leader. If $L(C_{sl}) - s > L(C_l)$ the suffix is a part of the slave, if $L(C_{sl}) - s < L(C_l)$ it is a part of the leader. When $L(C_{sl}) - s = L(C_l)$ no suffix is present.

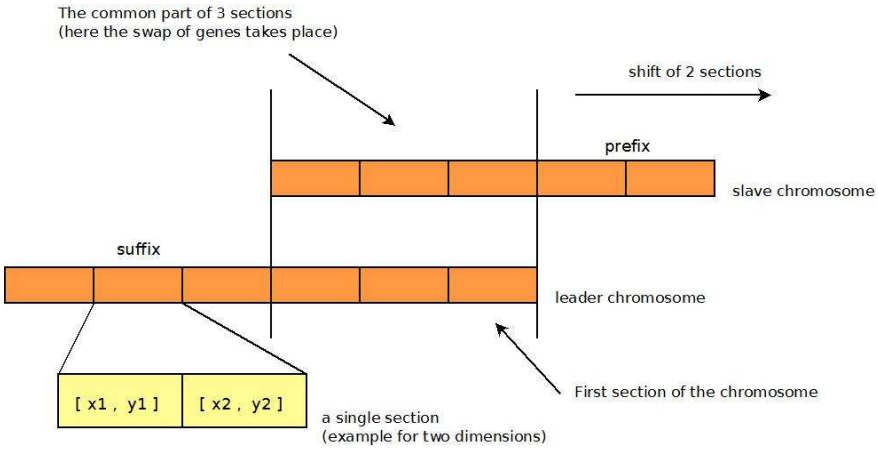

Fig. 2. The shift value and the result of chromosomes placement

2. The crossing point r is randomly (with a uniform distribution) selected within the area of overlapping parts of leader and slave chromosomes. It is picked as a natural number from range $[1, \min\{2d(L(C_{sl}) - s), 2dL(C_l)\}]$, where C_{sl} is a slave chromosome and C_l is a leader chromosome. The upper bound of value r comes from the fact that a slave chromosome can be longer than a leader one even after substraction of the prefix. Having the r value, the genes swap is conducted in a way that is presented in Fig. 3. The genes are swapped starting from the beginning of a leader chromosome and $s + 1$ section of a slave chromosome.

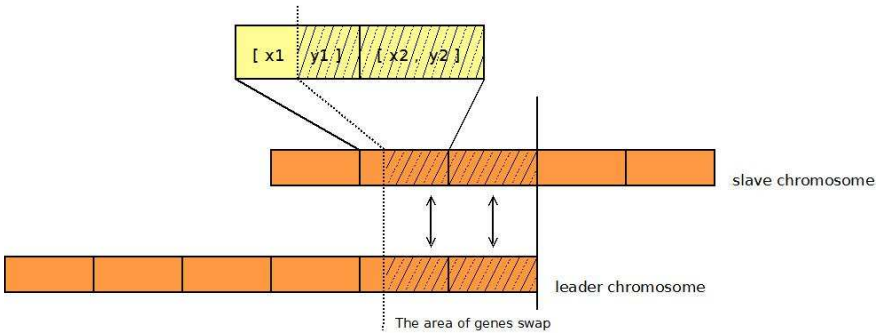

Fig. 3. The example of genes swap in a crossing-over process in two dimensional space, r = 7, shift = 2

3. After swapping the genes two overlapping parts of chromosomes are extracted and their adaptation is calculated. The one with the higher value is chosen as the first candidate for a descendant.
4. The sequence chosen in the previous point is combined with the prefix and suffix, which constitutes the second candidate. Note that the first and the second candidate can be the same sequence only if $L(C_{sl}) = L(C_l)$ and $s = 0$.
5. From two selected candidates the one having the higher value of adaptation is put in the next population both with the better (with respect to the adaptation) of its parents $L(C_{sl}), L(C_l)$.

2.3. Mutation

After having a new population as a result of selection and crossing-over, a mutation is conducted for each individual. In this algorithm two kinds of mutation are proposed. The first one is a classic mutation of single genes. The effect is that a single gene acquires a new value randomly selected with a uniform distribution from the domain of the encoded coordinate. As a result, one corner of a cube is moved randomly along a single axis which can have a great impact on a size and placement of a cube. This operation is performed with some small probability for each gene of a chromosome.

The second type is the mutation of a chromosome's length. The changes can result in either adding or substracting a random number of sections from the chosen chromosome C. In the first step value q is randomly selected with the normal distribution (0, 1). Then q is converted to its floor. Having this value we conduct the following operations.

1. If q is equal to 0 no changes are made to the chromosome.
2. If q is lower than 0 then it is checked if its absolute value is less than $L(C)-2$. If not, no changes are made to the chromosome as it would result in having a chromosome shorter than two sections. If that condition is satisfied, q sections are removed from the end of the chromosome C.
3. If q is has a positive value adding new sections takes place. The number of q sections are randomly generated and added to the end of the chromosome C. As a result of that operation the length of the chromosome is increased by $2qd$ new genes.

The length mutation is conducted with some probability for each whole chromosome. For that method to take effect on the chromosome the probability of conducting it should be significantly higher than probability of the classic mutation. It is suggested to take probability which is ten times higher than probability of the classic mutation.

2.4. The adaptation function

The adaptation function evaluates the placement and the size (volume) of the cubes encoded by a chromosome with reference to the clustered set. Its value is higher for those chromosomes, which cubes more accurately mark (localize) dense regions in the clustered set. The calculations are based on two main values – the number of elements of the set that are placed inside all cubes encoded by a chromosome (T) and the sum of the cubes' sizes (S) calculated with formula (1). Also a number of the cubes that do not surround any of the points of the clustered set (E) will be used in calculations.

$$S = \sum_{i=1}^{k} \prod_{j=1}^{d} |x^i_{2_j} - x^i_{1_j}|. \tag{1}$$

The value S is normalized by the function (2), where m is the cardinality of the clustered set and a is the size (volume) of a cube determined by the domain of the set. Value a can be interpreted as a volume of the minimal cube that would surround all the points in the clustered set and can be calculated with formula (3), where P is the clustered set with elements of the form $\left\{\left(p^1_1, p^1_2, ..., p^1_d\right), ..., (p^m_1, p^m_2, ..., p^m_d)\right\}$.

$$f(s) = \frac{mS}{a}. \tag{2}$$

$$a = \prod_{i=1}^{d} (\max\{p_i; p_i \in P\} - \min\{p_i; p_i \in P\}). \tag{3}$$

Finally, the adaptation function has the form

$$A(C) = \frac{T - f(S)}{E + 1}. \tag{4}$$

In formula (4) the E value acts as a punishment for these chromosomes that encode cubes that do not mark any points. Such individuals are highly undesirable and thus obtain a significantly lowered value of the adaptation. The role of parameters m and a in (2) is to make the adaptation function scalable with respect to the number of the elements in a clustered set, size of its domain and dimensions number.

The observation of the algorithm leads to the conclusion that with the growing number of the elements of a clustered set the value of T also grows up. In that case, without parameter m in (2), the role of the cubes volume would be diminished. It is so, because with the high value of T even significant changes of C would lead to comparably small changes in the value of the adaptation. According to the observations, such case leads to undesirable growth of the cubes and, as a result, to lowered precision of a clustering.

The use of parameter a in function (2) makes the adaptation function scalable in regard of both the size of a clustered set's domain and the number of dimensions. With the greater size of a domain the volume of the cubes encoded by a chromosome is also higher. What is more, a dimensions number plays very significant role in the value of (3) which grows exponentially with d. The normalization restricts the value of $\frac{S}{a}$ to the range [0, 1] and, as a result, value of (3) to [0, m]. Of course the number of points covered by a chromosome's cubes (T) also ranges between 0 and m. That observations lead to the conclusion that values T and $f(C)$ are comparable regardless of the values of m, a and d.

3. Tests

3.1. Data sets

The following tests were conducted for two sets of sample data. Both sets were generated in two dimensional Euclidean space. In both cases the domain was limited to $[0, 15] \times [0, 15] \subset \mathbb{R}^2$.

The first set (Fig. 4) consists of two well separated concentrations of randomly generated points. Each group is limited to the rectangle having the edge length of 5. The number of points is 40 (20 points in each group). The second set (Fig. 5) is constructed in such a way that two groups of points are combined with a thin path of points. The overall number of points in this case is 142 (54 points in each

condensed group and 34 for the path). It should be stressed that data having such structure have great practical importance, among others in intelligent monitoring of rotational machines, for instance wind turbines [1, 2].

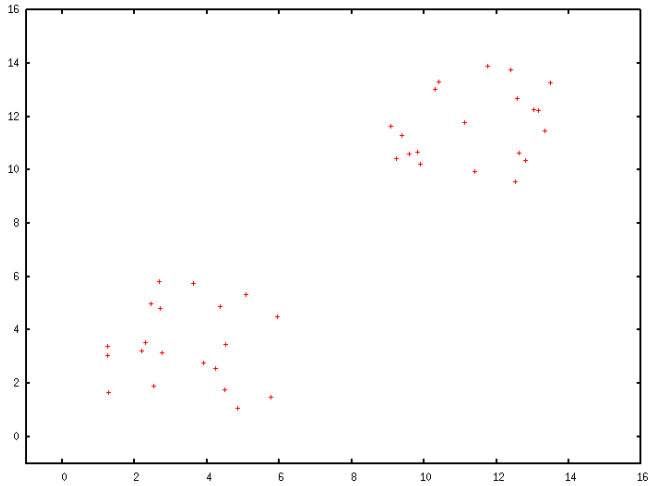

Fig. 4. Test data 1

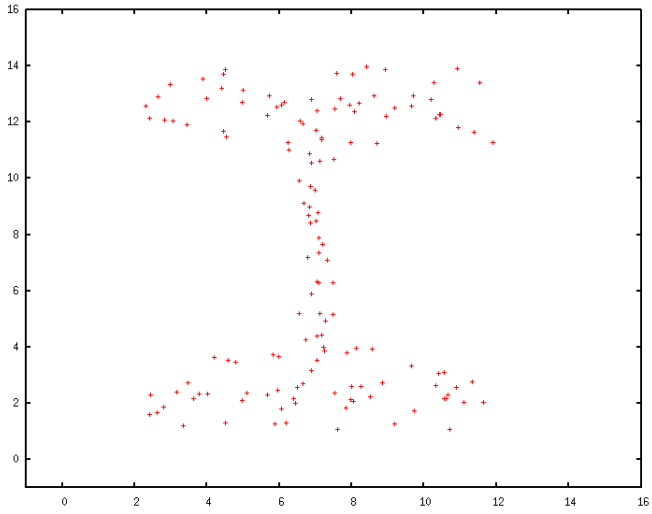

Fig. 5. Test data 2

3.2. Tests results

Tests were conducted with the population size of 25 chromosomes. The initial population was generated randomly with the length of the individuals (number of sections) randomly generated with the normal distribution (5, 1) (median of the length was 5). The probabilities of simple mutation and length mutation are 0.02. The probability of performing crossing-over for two selected chromosomes is 0.95.

Figures 6, 7 and 8 show results of the algorithm execution for the first sample data (see Fig. 4).

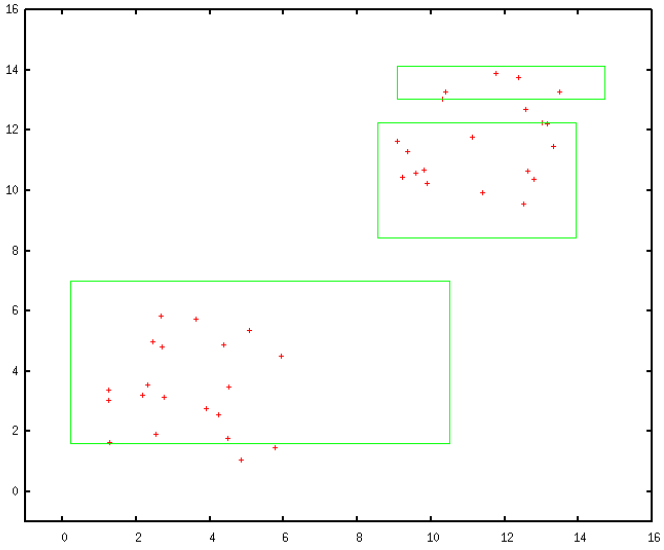

Fig. 6. The result after 30 iterations (test data 1)

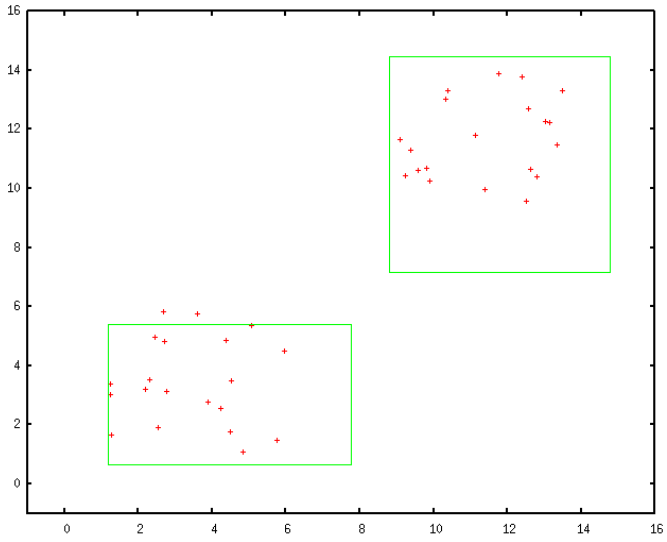

Fig. 7. The result after 60 iterations (test data 1)

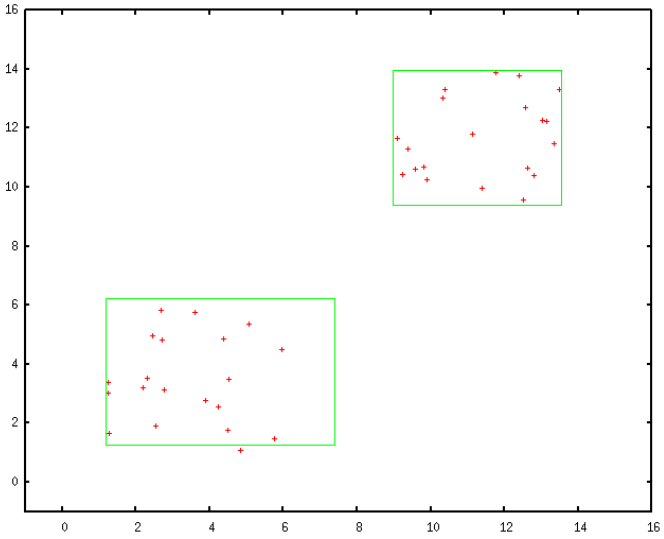

Fig. 8. The result after 150 iterations (test data 1)

The following results were obtained for the second data set (see Fig. 5).

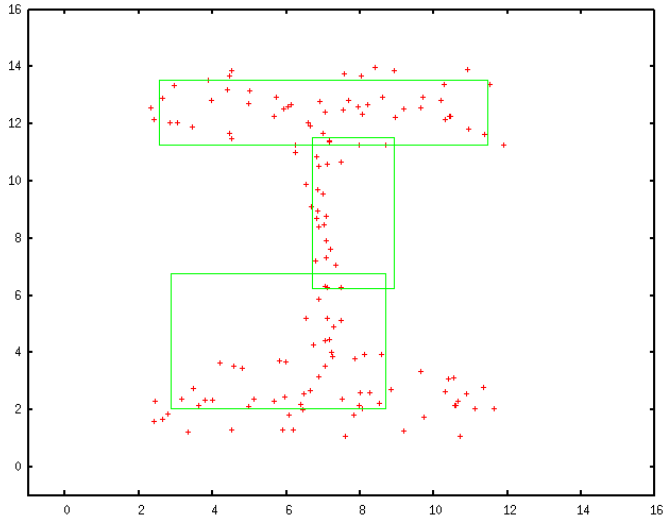

Fig. 9. The result after 30 iterations (test data 2)

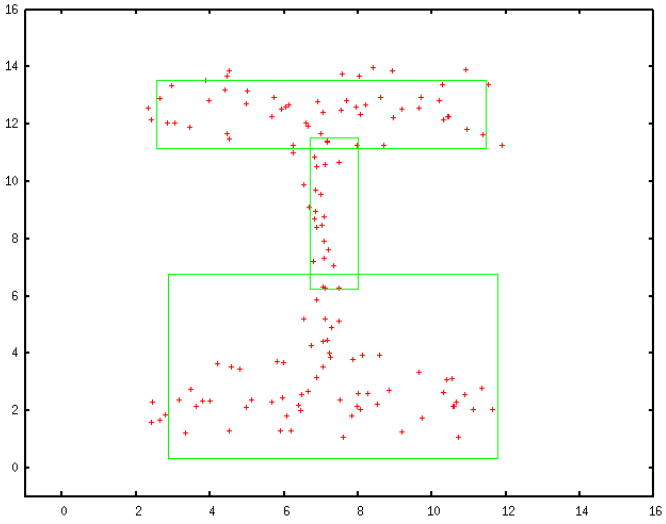

Fig. 10. The result after 60 iterations (test data 2)

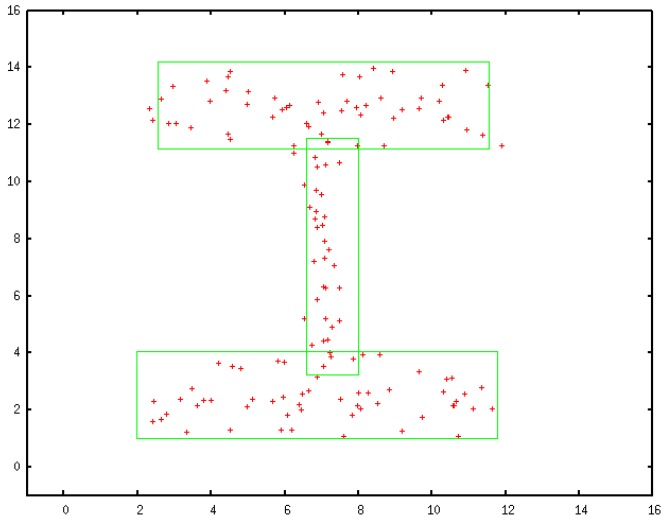

Fig. 11. The result after 150 iterations (test data 2)

4. Concluding remarks

Test results show good performance of the algorithm for the medium size of the population (25 chromosomes). In both cases, after 60 iterations the result was fairly close to the intuitive result of clustering, based on the observation of two dimensional data.

The described algorithm is very scalable considering the memory size. It is able to localize separated dense regions that can be interpreted as clusters. The algorithm uses cubes for marking the separated clusters. The adaptation is estimated with respect to those cubes' placement and size as well as the clustered set's elements arrangement. As a result of using cubes, which are convex solids, the algorithm shows better performance for those clusters that have convex shapes and are well separated by the areas of lower density.

5. References

[1] Barszcz T., Bielecki A., Wójcik M.; *ART-type artificial neural networks applications for classification of operational states in wind turbines*, Lecture Notes in Artificial Intelligence, 6114, 2010, pp. 11–18.

[2] Barszcz T., Bielecka M., Bielecki A., Wójcik M.; *Wind turbines states classification by a fuzzy-ART neural network with a stereographic projection as a signal normalization*, Lecture Notes in Computer Science, 6594, 2011, pp. 225–234.

[3] Bielecki A., Bielecka M., Chmielowiec A.; *Input signals normalization in Kohonen neural networks*, Lecture Notes in Artificial Intelligence, 5097, 2008, pp. 3–10.

[4] Cavicchio D.J.; *Adaptative search using simulated evolution*, PhD thesis, University of Michigan, 1970.

[5] Goldberg D.E.; *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston 1989.

[6] Goswami G., Liu J.S., Wong W.H.; *Evolutionary Monte Carlo methods for clustering*, Journal of Computational and Graphical Statistics, 16, 2007, pp. 855–876.

[7] Hruschka E.R., Ebecken N.F.F.; *A genetic algorithm for cluster analysis*, Intelligent Data Analysis, 7, 2003, pp. 15–25.

[8] Lorena L.A.N., Furtado J.C.; *Constructive genetic algorithm for clustering problems*, Evolutionary Computation, 9, 2001, pp. 309–328.

Received June 11, 2010